TRANSMITTAL OF APPEAL BRIEF (Large Entity)

Docket No.
ITL.1482US

In Re Application Of: **Michael Y. Lai**

Application No.	Filing Date	Examiner	Customer No.	Group Art Unit	Confirmation No.
10/674,364	**September 29, 2003**	**Isaac Tuku Tecklu**	**47795**	**2192**	**5221**

Invention: **Method and Apparatus for Bit Field Optimization**

COMMISSIONER FOR PATENTS:

Transmitted herewith is the Appeal Brief in this application, with respect to the Notice of Appeal filed on:

November 21, 2007

The fee for filing this Appeal Brief is: **$510.00**

☒ A check in the amount of the fee is enclosed.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account.

☒ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. **20-1504**. I have enclosed a duplicate copy of this sheet.

☐ Payment by credit card. Form PTO-2038 is attached.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

Signature

Dated: **January 17, 2008**

Timothy N. Trop, Reg. No. 28,994
TROP, PRUNER & HU, P.C.
1616 S. Voss Road, Suite 750
Houston, TX 77057
713/468-8880 [Phone]
713/468-8883 [Fax]

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to "Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450" [37 CFR 1.8(a)] on

January 17, 2008
(Date)

Signature of Person Mailing Correspondence

Nancy Meshkoff

Typed or Printed Name of Person Mailing Correspondence

cc:

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Applicant:	Michael Y. Lai	Art Unit:	2192
Serial No.:	10/674,364	Examiner:	Isaac Tuku Tecklu
Filed:	September 29, 2003	Atty Docket:	ITL.1482US (P16116)
For:	Method and Apparatus for Bit Field Optimization	Assignee:	Intel Corporation

Mail Stop **Appeal Brief-Patents**
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Date of Deposit: January 17, 2008
I hereby certify under 37 CFR 1.8(a) that this correspondence is being deposited with the United States Postal Service as **first class mail** with sufficient postage on the date indicated above and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Nancy Meshkoff

TABLE OF CONTENTS

REAL PARTY IN INTEREST

The real party in interest is the assignee Intel Corporation.

RELATED APPEALS AND INTERFERENCES

None.

STATUS OF CLAIMS

Claims 1-57 (Rejected).

Claims 1-57 are rejected and are the subject of this Appeal Brief.

STATUS OF AMENDMENTS

No amendments were made in the Reply to Final Rejection submitted on May 31, 2007. All amendments have therefore been entered.

SUMMARY OF CLAIMED SUBJECT MATTER

In the following discussion, the independent claims are read on one of many possible embodiments without limiting the claims:

1. A method comprising:

generating an intermediate representation (IR) of a source program, where the source program includes one or more instructions for processing data in a bit field within a data structure (Fig. 1, 104; spec. at para. 15, 16, and 21);

modifying the intermediate representation to more efficiently execute the one or more instructions for processing the bit field data (Fig. 1, 106; spec. at para. 23); and

generating resultant code based on the modified intermediate representation (Fig. 1, 108; spec. at para. 24).

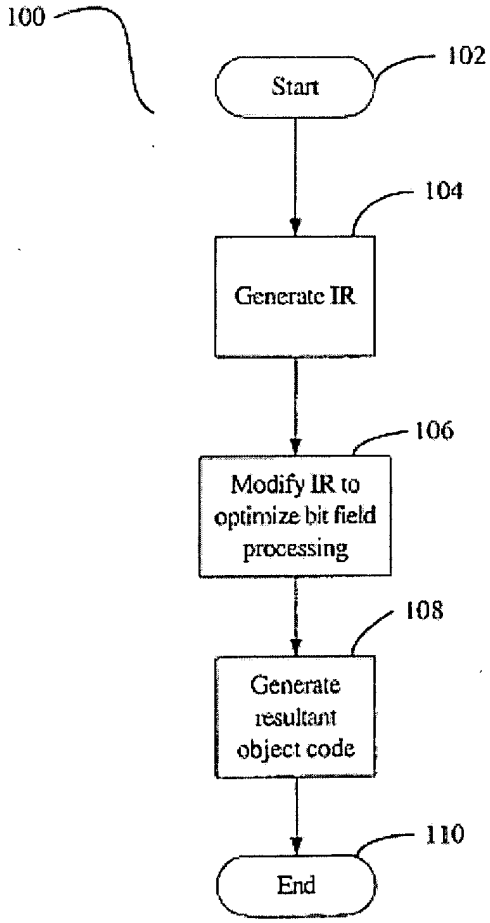

Fig. 1

23. An article comprising:

a machine-readable storage medium having a plurality of machine accessible instructions, which if executed by a machine, cause the machine to perform operations comprising:

generating an intermediate representation (IR) of a source program, where the source program includes one or more instructions for processing data in a bit field within a data structure (Fig. 1, 104; spec. at p. 7, para. 15, 16, and 21);

modifying the intermediate representation to more efficiently execute the one or more instructions for processing the bit field data (Fig. 1, 106; spec. at para. 23); and

generating resultant code based on the modified intermediate representation (Fig. 1, 108; spec. at para. 24).

[CONTINUED ON NEXT PAGE]

45. A computer readable medium storing instructions to cause a computer to optimize the processing of bit fields, said instructions comprising:

a front end (Fig. 2, 230) to generate an intermediate representation of a source program (spec. at para. 26);

an optimizer (Fig. 2, 235) to modify the intermediate representation (IR) to provide for optimized processing of one or more bit fields (spec. at para. 27); and

a back end (Fig. 2, 240) to generate resultant code based on the modified intermediate representation (spec. at para. 29).

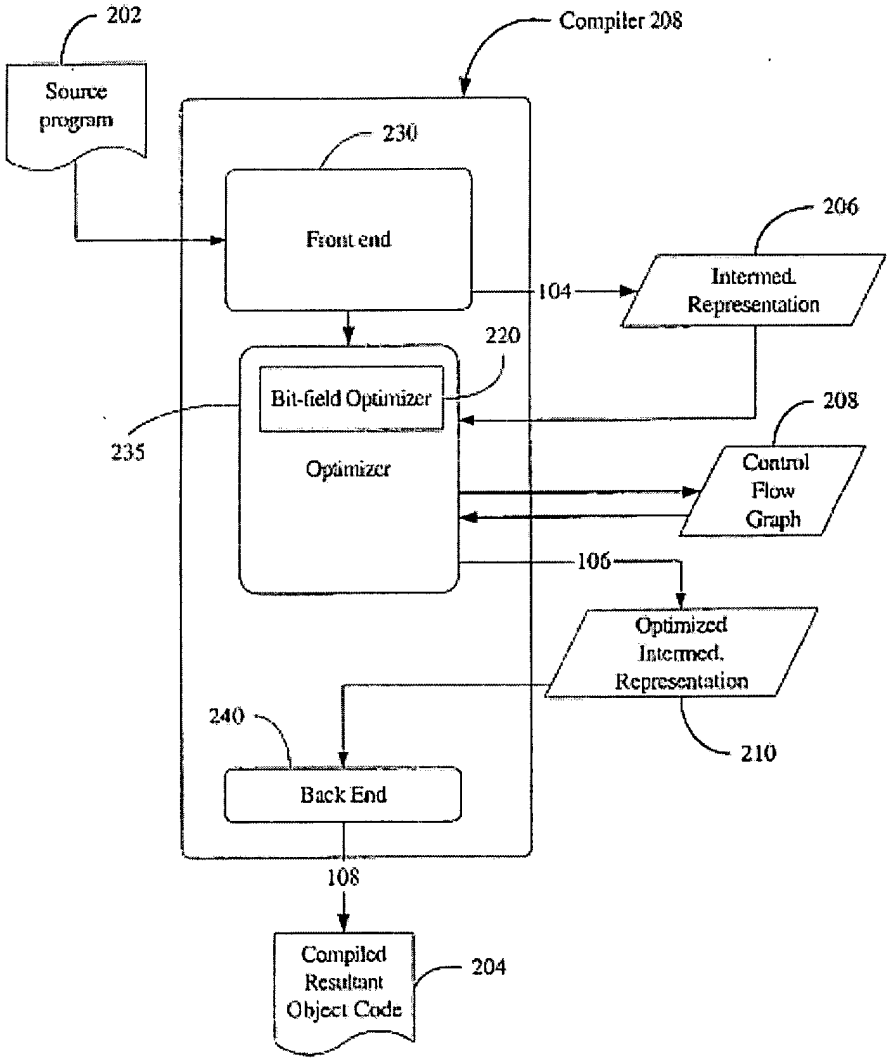

Fig. 2

At this point, no issue has been raised that would suggest that the words in the claims have any meaning other than their ordinary meanings. Nothing in this section should be taken as an indication that any claim term has a meaning other than its ordinary meaning.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. **Whether claims 1-57 are anticipated under 35 U.S.C. § 102(b) by Tye (US 6,226,789).**

ARGUMENT

A. Are claims 1-57 anticipated under 35 U.S.C. § 102(b) by Tye (US 6,226,789)?

Claim 1 calls for generating an intermediate representation of a source program where the source program includes one or more instructions for processing data in a bit field within a data structure. The claim further calls for modifying that intermediate representation "to more efficiently execute the one or more instructions for processing the bit field data."

The final office action suggests that bit fields are taught and that somehow the source program includes instructions for processing data in the bit field. If we accept that for argument purposes, we are left with the question -- where is the modifying that intermediate representation to more efficiently execute the one or more instructions for processing the bit field data?

The asserted support for this limitation consists entirely of the clause "... then modifies the IR to produce a final version of the IR that corresponds to instructions in the second instruction set." See Tye at column 63, lines 20-22. But this says nothing about the bit field data whatsoever. Not only does it not even mention the bit field data in particular, but it in no way suggests that the intermediate representation is modified "to more efficiently execute the one or more instructions for processing the bit field data." Instead, it is indicated that the intermediate representation is modified to produce a final version "that corresponds to instructions in the second instruction set."

There is no basis for the conclusion drawn in the office action that this language somehow suggests causing the intermediate representation to be modified to more efficiently execute the one or more instructions for processing the bit field data. The bit field data is not even mentioned with respect to the modification. This is an example of the reference teaching the general, not the specific. Simply teaching modification does not teach the claimed modification.

* * *

Applicant respectfully requests that each of the final rejections be reversed and that the claims subject to this Appeal be allowed to issue.

Respectfully submitted,

Date: January 17, 2008

Timothy N. Trop, Reg. No. 28,994
TROP, PRUNER & HU, P.C.
1616 S. Voss Road, Suite 750
Houston, TX 77057
713/468-8880 [Phone]
713/468-8883 [Fax]

CLAIMS APPENDIX

The claims on appeal are:

1. A method comprising:

generating an intermediate representation (IR) of a source program, where the source program includes one or more instructions for processing data in a bit field within a data structure;

modifying the intermediate representation to more efficiently execute the one or more instructions for processing the bit field data; and

generating resultant code based on the modified intermediate representation.

2. The method of claim 1, wherein modifying the intermediate representation further comprises:

pre-processing the IR to perform preliminary modification of the IR.

3. The method of claim 2, wherein modifying performing pre-processing further comprises:

performing data flow analysis to gather information regarding definition and usage of the bit field data; and

generating a graph to classify the information.

4. The method of claim 3, wherein generating a graph further comprises:

generating a graph to classify the information in relation to an associated packet.

5. The method of claim 2, wherein modifying the intermediate representation further comprises:

(a) allocating a temporary variable to hold the bit field data; and

(b) modifying the IR so that the temporary variable is processed in accordance with the instructions.

6. The method of claim 5, further comprising:

(c) assigning the value of the temporary variable to a memory.

7. The method of claim 6, further comprising:

performing steps (a), (b) and (c) for a single basic block.

8. The method of claim 7, further comprising:

identifying two or more sub-blocks within the basic block.

9. The method of claim 8, wherein:

steps (a) , (b) and (c) are performed for each sub-block.

10. The method of claim 5, further comprising:

determining whether all of the one or more instructions for processing the bit field data are read-after-write instructions; and

performing steps (a) and (b) only if the determination is false.

11. The method of claim 6, further comprising:

determining whether any of the one or more instructions for processing the bit field data are write instructions; and

performing step (c) only if the determination is true.

12. The method of claim 6, further comprising:

removing the modifications effected by steps (a), (b) and (c) upon determining that such removal is expected to provide an efficiency benefit in the resultant code.

13. The method of claim 2, wherein pre-processing further comprises:

disambiguating a memory reference to the bit field.

14. The method of claim 1, wherein modifying the intermediate representation further comprises:

modifying the IR so that multiple instructions to initialize respective bit fields of a data structure are performed with a single write to a memory.

15. The method of claim 14, wherein the multiple instructions occur within a pre-defined maximal scope.

16. The method of claim 1, wherein modifying the intermediate representation further comprises:

modifying the IR so that multiple read instructions for respective bit fields of a data structure are performed with a single read from a memory.

17. The method of claim 16, wherein the multiple read instructions occur within a pre-defined maximal scope.

18. The method of claim 1, wherein modifying the intermediate representation further comprises:

modifying the IR so that multiple write instructions to respective bit fields of a data structure are performed with a single write to a memory.

19. The method of claim 18, wherein the multiple read instructions occur within a pre-defined maximal scope.

20. The method of claim 1, wherein modifying the intermediate representation further comprises:

determining that a first instruction, being one of the one or more instructions, indicates a bit-wise logical operation on the bit field data;

determining that a second instruction of the source program indicates a bit-wise logical operation on a second bit field within the data structure; and

modifying the IR so that the first and second instructions are performed via a single read from a memory.

21. The method of claim 20, wherein the bit-wise logical operation is a bit-wise OR operation.

22. The method of claim 20, wherein the bit-wise logical operation is a bit-wise AND operation.

23. An article comprising:

a machine-readable storage medium having a plurality of machine accessible instructions, which if executed by a machine, cause the machine to perform operations comprising:

generating an intermediate representation (IR) of a source program, where the source program includes one or more instructions for processing data in a bit field within a data structure;

modifying the intermediate representation to more efficiently execute the one or more instructions for processing the bit field data; and

generating resultant code based on the modified intermediate representation.

24. The article of claim 23, wherein the instructions that cause the machine to modify the intermediate representation further comprise instructions that cause the machine to:

perform preliminary modification of the IR.

25. The article of claim 24, wherein the instructions that cause the machine to modify the intermediate representation further comprise instructions that cause the machine to:

gather information regarding definition and use of the bit field data; and

generate a graph to classify the information.

26. The article of claim 25, wherein the instructions that cause the machine to generate a graph further comprise instructions that cause the machine to:

generate a graph to classify the information in relation to an associated packet.

27. The article of claim 24, wherein the instructions that cause the machine to modify the intermediate representation further comprise instructions that cause the machine to:

(a) allocating a temporary variable to hold the bit field data; and

(b) modifying the IR so that the temporary variable is processed in accordance with the instructions.

28. The article of claim 27, further comprising a plurality of machine accessible instructions, which if executed by a machine, cause the machine to perform operations comprising:

(c) assigning the value of the temporary variable to a memory.

29. The article of claim 28, further comprising a plurality of machine accessible instructions, which if executed by a machine, cause the machine to perform operations comprising:

performing steps (a), (b) and (c) for a single basic block.

30. The article of claim 29, further comprising a plurality of machine accessible instructions, which if executed by a machine, cause the machine to perform operations comprising:

identifying two or more sub-blocks within the basic block.

31. The article of claim 30, further comprising a plurality of machine accessible instructions, which if executed by a machine, cause the machine to perform operations comprising:

performing steps (a), (b) and (c) for each sub-block.

32. The article of claim 27, further comprising a plurality of machine accessible instructions, which if executed by a machine, cause the machine to perform operations comprising:

determining whether all of the one or more instructions for processing the bit field data are read-after-write instructions; and

performing steps (a) and (b) only if the determination is false.

33. The article of claim 28, further comprising a plurality of machine accessible instructions, which if executed by a machine, cause the machine to perform operations comprising:

determining whether any of the one or more instructions for processing the bit field data are write instructions; and

performing step (c) only if the determination is true.

34. The article of claim 28, further comprising a plurality of machine accessible instructions, which if executed by a machine, cause the machine to perform operations comprising:

removing the modifications effected by steps (a), (b) and (c) upon determining that such removal is expected to provide an efficiency benefit in the resultant code.

35. The article of claim 24, wherein the instructions that cause the machine to perform preliminary modification of the IR further comprise instructions that cause the machine to:

disambiguate a memory reference to the bit field.

36. The article of claim 23, wherein the instructions that cause the machine to modify the intermediate representation further comprise instructions that cause the machine to:

modify the IR so that multiple instructions to initialize respective bit fields of a data structure are performed with a single write to a memory.

37. The article of claim 36, wherein the multiple instructions occur within a pre-defined maximal scope.

38. The article of claim 23, wherein the instructions that cause the machine to modify the intermediate representation further comprise instructions that cause the machine to:

modify the IR so that multiple read instructions for respective bit fields of a data structure are performed with a single read from a memory.

39. The article of claim 38, wherein the multiple read instructions occur within a pre-defined maximal scope.

40. The article of claim 23, wherein the instructions that cause the machine to modify the intermediate representation further comprise instructions that cause the machine to:

modify the IR so that multiple write instructions to respective bit fields of a data structure are performed with a single write to a memory.

41. The article of claim 40, wherein the multiple read instructions occur within a pre-defined maximal scope.

42. The article of claim 23, wherein the instructions that cause the machine to modify the intermediate representation further comprise instructions that cause the machine to:

determine that a first instruction, being one of the one or more instructions, indicates a bit-wise logical operation on the bit field data;

determine that a second instruction of the source program indicates a bit-wise logical operation on a second bit field within the data structure; and

modify the IR so that the first and second instructions are performed via a single read from a memory.

43. The article of claim 42, wherein the bit-wise logical operation is a bit-wise OR operation.

44. The article of claim 42, wherein the bit-wise logical operation is a bit-wise AND operation.

45. A computer readable medium storing instructions to cause a computer to optimize the processing of bit fields, said instructions comprising:

a front end to generate an intermediate representation of a source program;

an optimizer to modify the intermediate representation (IR) to provide for optimized processing of one or more bit fields; and

a back end to generate resultant code based on the modified intermediate representation.

46. The medium of claim 45, wherein:

the optimizer includes a pre-processor to perform preliminary processing of the intermediate representation.

47. The medium of claim 46, wherein:

the pre-processor includes a data flow analyzer to perform data flow analysis and to generate a graph.

48. The medium of claim 46, wherein:

the pre-processor includes a registerizer to modify the intermediate representation to allocate a temporary variable for a bit field variable used in the source program.

49. The medium of claim 47, further comprising:

an unregisterizer to selectively reverse the modification performed by the registerizer.

50. The medium of claim 45, wherein:

the optimizer includes a bit-specific optimizer to modify the IR such that processing of bit fields indicated by the source program is more efficient.

51. The medium of claim 50, wherein:

the bit-specific optimizer includes an aggregate initializer to initialize multiple bit fields within a data structure via a single write to memory.

52. The medium of claim 50, wherein:

the bit-specific optimizer includes a read/write combiner to read multiple bit fields within a data structure via a single read from memory.

53. The medium of claim 52, wherein:

the read/write combiner is further to initialize write bit fields within a data structure via a single write to memory.

54. The medium of claim 50, wherein:

the bit-specific optimizer includes a juxtaposition merger to determine that a first instruction, being one of the one or more instructions, indicates a bit-wise logical operation on the bit field data;

the juxtaposition merger further to determine that a second instruction of the source program indicates a bit-wise logical operation on a second bit field within the data structure; and

the juxtaposition optimizer further to modify the IR so that the first and second instructions are performed via a single read from a memory.

55. The medium of claim 50, wherein:

the bit-specific optimizer includes an "or" optimizer to merge logical "or" statements of a conditional statement together such that they are executed via a single read statement.

56. The medium of claim 55, wherein:

the "or" optimizer is further to merge bit-wise "or" statements of a conditional statement together such that they are executed via a single read statement.

57. The medium of claim 50, wherein:

the bit-specific optimizer includes an "and" optimizer to merge logical "and" statements of a conditional statement together such that they are executed via a single read statement.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.